项目报告

施云翀，马嘉欣，徐奕

2025年1月

1 引言

本项目为SI100B课程的最终项目，项目文件上传至：<https://github.com/ProjectEggGame/PikyorEgg.git>。

这是一款任务闯关类的养成游戏。玩家将作为一只被主人抛弃的小鸡醒来，通过与世界的交互，完成一系列任务，最终完成孵出小鸡、繁育后代的目标，这个游戏希望能通过沉浸式的体验让玩家感受到成长和生育的美好，从而增加我国生育率。

在主世界中，你可以通过捡拾米粒、收集树枝来完成任务，并通过转送门进入老巫婆鸡的世界接受教育，在这个过程中，要时刻提防蠢蠢狐狸的袭击，通过回答问题寻找线索，避免坏长老鸡的攻击，找到真正的老巫婆鸡。

为了实现下蛋的任务，你需要与情敌母鸡战斗，为自己的孩子找到一只帅气公鸡当爸爸，接受他的受精。

最终，你进入蛋蛋工厂，下一个蛋并孵化你的宝宝出生。

2 项目实施

2.1 场景

2.1.1 DynamicWorld主世界

游戏的主世界，小鸡在此完成大部分的任务。

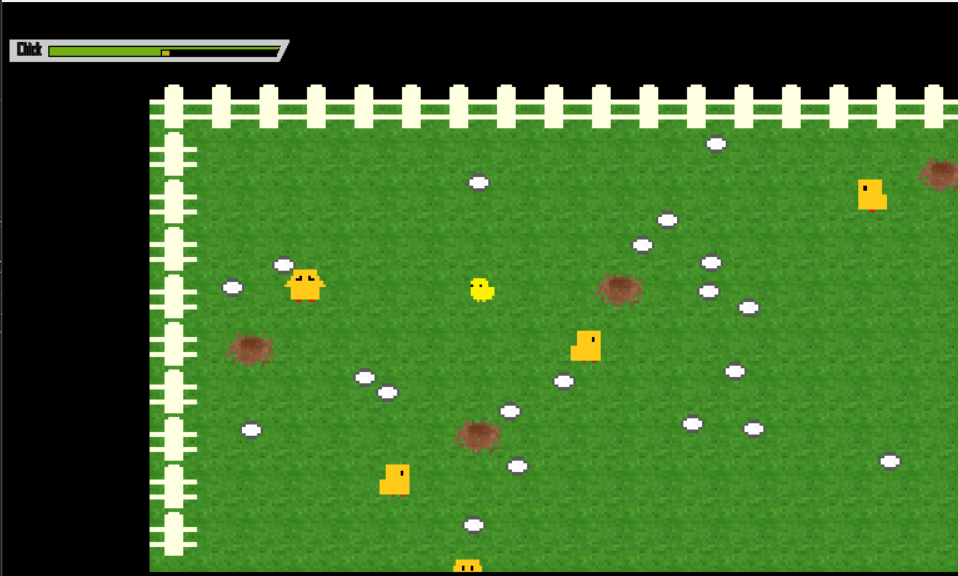
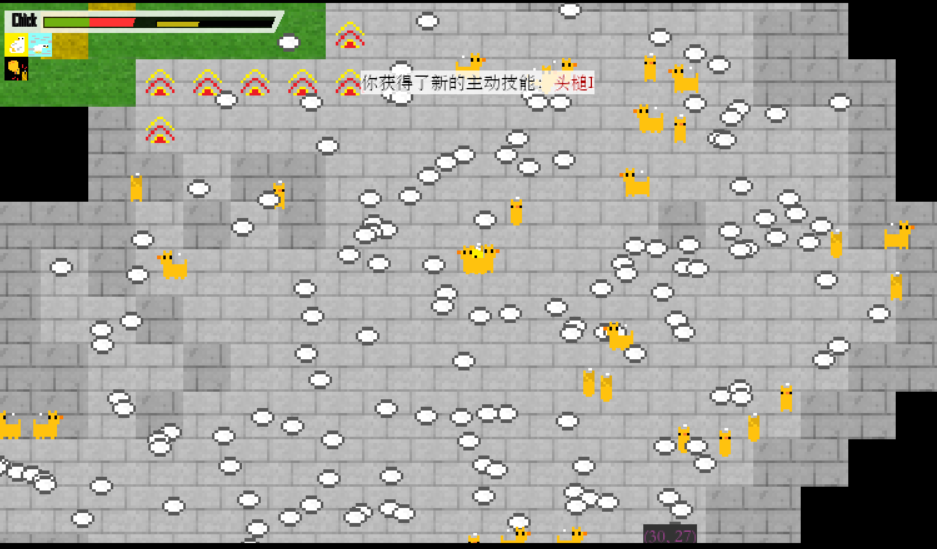
  

图1、2、3 DynamicWorld部分展示

游戏地图随机生成。创建地图前，玩家可选择决定地图生成的随机数种子，然后使用python的random库生成随机数实现地图随机生成。玩家恒定出生于(0, 0)，随机生成两个方向向量和长度，生成两段草径，并在第一个方向向量的末端生成一个狐狸塔、第二个方向向量末端固定生成一个大鸡圈。地图中的狐狸、米粒、母鸡等均在游戏开始时由随机数决定位置，游戏过程中由随机数决定随机游走方向。

2.1.2 WitchWorld异世界

老巫婆的世界中善恶并存，唯有得到长老教化的小鸡方能诞下小皇鸡。老巫婆鸡可以帮助主角小鸡接受教育，但是前提是你要躲避狐狸的追踪和袭击，寻找线索并分辨出假长老鸡。

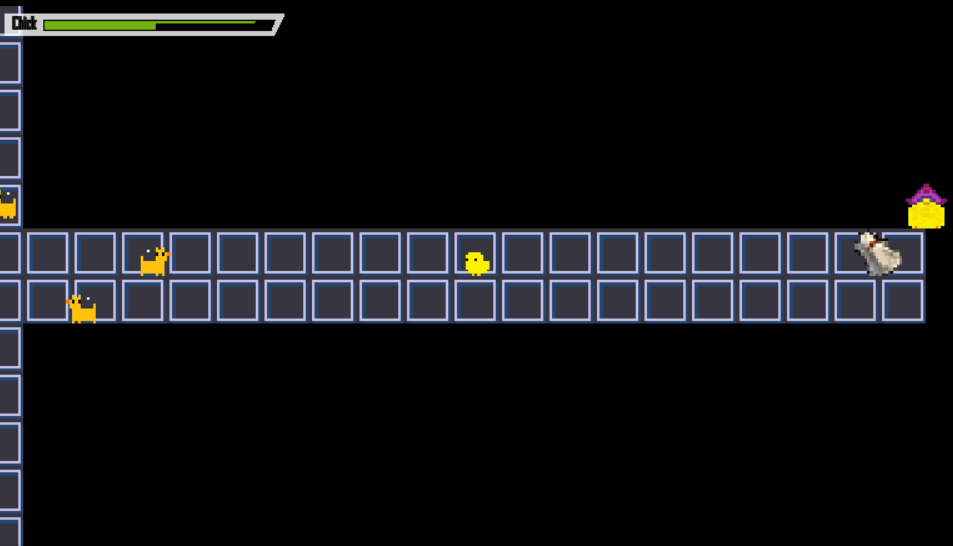
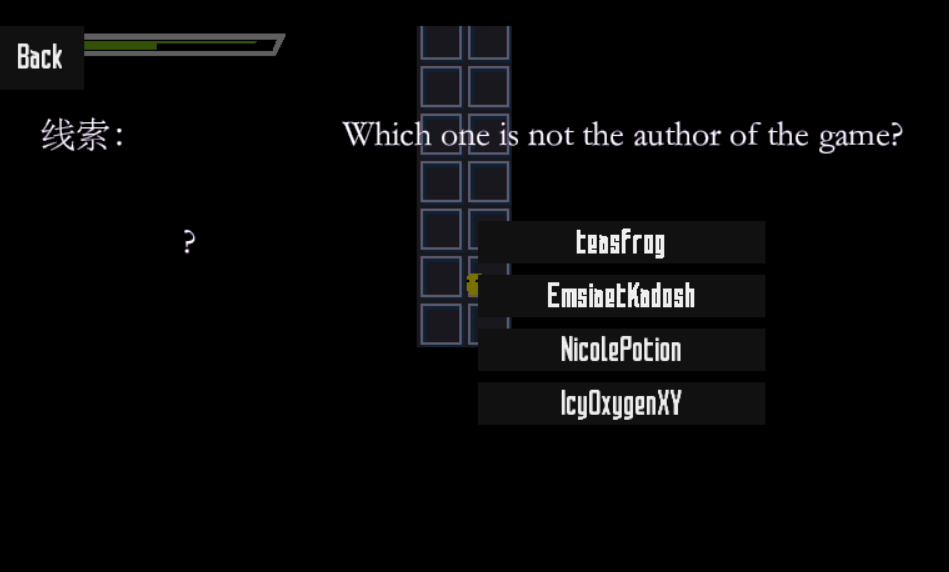
 

图1、2、3 WitchWorld部分展示

* + 1. 场景镜头移动

所有World类世界（即上述两场景）可实现镜头跟随，并且我们额外实现了：

1）按下空格键，镜头跟随关闭

2）按住鼠标中键滚轮可以拖动地图。

镜头在游戏中以一个向量存储其位于游戏地图上的位置。渲染时，根据地图缩放比例，仅筛选出位于窗口可视范围内的实体和方块渲染，每个元素通过自身位置、镜头偏移和地图缩放比例，计算出在屏幕上的位置，随后渲染。

2.1.3 互动物品

互动物品在World存在，有多种互动物品，照片展示如下。

（差张树枝图片）    

图 米粒 图 树枝 图 鸡窝 图 传送门 图 线索 图 神奇的墙 图 栅栏

1)米粒：Entity类，当主角与米粒距离小于特定值后，米粒被自动捡拾，并且被从世界移除，米粒数量会转化为小鸡（玩家）的生命值。

2）树枝：Entity类，与米粒几乎一样。树枝数量会转化为玩家背包中的树枝数量。

3）鸡窝：Entity类，收集100树枝后，玩家可在家中键入H建造鸡窝。

4）传送门：Block类，当玩家在该方块上停留满3秒，传送启动，玩家会被传送到异世界。

5）线索：Entity类，类似米粒，但不会改变小鸡属性，只会调出一个Window窗口，并且被从世界移除。

6）神奇的墙：Block类，叠在Map的Block上，允许玩家通过，Enemy不可通过。

7）栅栏：Block类，叠在Map的Block上，谁都不能通过，装饰的作用更大。

交互实现说明：

实体被创造后，其tick函数会在每一帧被调用，该函数中有实体坐标和玩家坐标的距离计算，并检测是否小于特定值，若满足，则执行对应操作，如调用player中的grow函数来实现吃了一粒米后玩家生命值1点的增长。

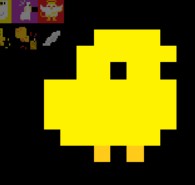
#收集足够的树枝后，玩家可以在家园中通过按下H键建造一个属于自己的鸡窝。

玩家通过努力，可以下一个属于自己的蛋。

2.2 角色

2.2.1 主要角色

小鸡！主要角色是由玩家控制的小鸡。



移动控制：

玩家通过wasd四个按键可以操控小鸡进行移动，其中四个方向的走动配有不同的动画。小鸡拥有自己的坐标，因为最后根据坐标被渲染到Map上，所以我们按键的控制本质是对小鸡x、y坐标做增量加减，每一帧检测按键是否按下，按下则执行加减。此外，同时按下Ctrl和方向我们使加减的增量翻倍，同时按下Shift和方向，我们使加减的增量减半。

小鸡可以通过传送门进入不同场景，通过碰撞实现拾取与交互。主要角色的移动通过检测按键的按下与抬起的状态来实现，当按键被按下时，角色就会向指定方向移动，按键抬起移动停止。（按键实现方法）

（详细解释）

小鸡属性：

小鸡有自己的生命值、背包树枝值、血条、技能包。展示在了每个World的左上角以及状态面板里（E键调出）



图 这张图既可以看到血条又能展示面板

2.2.2 友好NPC

2.2.2.1老巫婆鸡：

存在于异世界，我们让老巫婆鸡任意的移动、但极为缓慢，等待玩家答题得线索后找到它，确认并找到它后，它会作为传送门送小鸡回到主世界。这里主要通过坐标和random模块实现任意方向的移动，然后还是tick函数中的距离计算，检测玩家并传送。



图 老巫婆鸡

2.2.2.2 Strong的公鸡：

帅气公鸡是第二个友好NPC,在主世界中为主角的蛋提供一个爸爸。玩家要接受他的受精，完成第四个目标任务

2.2.3 敌人

2.2.3.1 蠢蠢的狐狸

蠢蠢狐狸是出现频率最高的敌人，他们守卫在主世界的多米区，也在老巫婆世界出现。蠢蠢狐狸具备实时战斗系统。

一样，还是tick函数检测玩家，一旦距离小于A，狐狸开始以玩家实时所在位置为目标位置开始移动，一旦距离小于B，调用player.damage()函数，实现对小鸡血量的减少。



图 蠢蠢的狐狸在盯着你看

2.2.3.2 丰腴的情敌

情敌母鸡也可以攻击玩家，且战斗力更强，攻击的能力同狐狸一样实现，仅设置了不同的参数，不过多赘述。

根据剧情，我们只有把母鸡杀了，小鸡才能上位。因此，进攻系统出现了。详细的进攻实现我们会放在创意当中介绍，这里介绍Enemy部分的被进攻相关部分：每只母鸡都有自己独立的hp血量，你可以通过将鼠标放在该鸡上看到血量剩余。血量的创建和加加减减写在Damageable类（继承Entity类）中，Player继承故能被伤害，Enemy类也继承，也支持被伤害。



图 丰腴的母鸡永远丰腴

公鸡和母鸡的形影不离的实现：

2.2.3.3 假长老鸡

剧情需要，假长老鸡作为“错误选项”的身份在异世界存在。他们的伤害不同于狐狸与母鸡，只会对玩家造成一次伤害，用bool类型的flag实现。



图 这些长老中存在老巫婆鸡哦

2.3 游戏机制

2.3.1 核心机制

玩家是一只被原主人抛弃的小鸡，它唯一的愿望就是长大并生下一颗厉害的蛋，也就是本游戏的终极目标——产蛋。

我们为终极下蛋任务做了铺垫和细化，变成了五个任务：吃米长大——捡枝造窝——得长老教化——觅得良配——许下对蛋的心愿。任务会逐一解锁，也需逐一完成，解锁全部任务，通关游戏并得到一颗独一无二的蛋 的图片，供玩家保存。

在世界窗口下，玩家键入Tab可调出任务面板。任务在未解锁的时候不能查看，上一项任务未完成，下一项任务就不会解锁。这里的任务页面切换，是由页面类的变量，等于不同数时渲染不同文字，来实现的——好用的if。



图 任务面板

作为一款养成类任务闯关游戏，游戏的目标就是我们为玩家提供了一个任务面板。玩家可以按下Tab键进行查看。玩家通过操作主要角色移动来与世界中的实体碰撞，碰撞后触发相应事件。玩家在世界中生存，完成任务面板提供的任务，并最终下一个自己的蛋。

2.3.2 碰撞系统

原设计案中，只检测运动起始点和终止点的碰撞是完全不严谨的，速度较快时极有可能穿墙而过；但检测方块路径移动碰撞会使代码过于复杂。为简化模型，我们将所有实体均视为点，多个实体之间的临近判定改为实体位置的曼哈顿距离判定；实体与方块的碰撞采用较为严谨的判定系统。

在World类中有一个rayTraceBlock函数，可以一次性获取由起始点开始、特定方向、特定长度内，所有经过的方块和与方块的碰撞点。遍历其返回列表，即可获得实体移动时碰撞到的方块。特别地，当实体撞到方块边界或方块角落时，如果有剩余的速度，游戏会将剩余速度投影到撞击后实体仍然能够移动的方向。

原始速度

实际速度

2.3.3 资源系统（？）

游戏的资源管理交由Texture类和ResourceManager类实施。ResourceManager存储了所有注册的Texture实例，避免了频繁的文件读写，每个Texture实例对应唯一ID即资源文件路径，并分别选择性地适应了三种渲染方式（Map、UI、System三种），针对不同需求，调用Texture类的对应渲染函数即可实现。需要自定义渲染时，也可以调用getSurface直接获取pygame表面进行相应的绘制工作。

2.4 LLM

2.4.1 对话系统

键入Enter即可跳出对话面板



此处我们让LLA语言系统作为一个“系统”存在，（感觉需要在最开始的plot最后强调一下 有任何难题可以像LLA询问）我们训练它说话简短，不回答与游戏无关的问题，并通过让它了解游戏的内容与玩法、以及所有键入的用法，来训练它成为一个系统。

在此界面中，单击输入框开始输入，输入框适应了中文输入法，可以使用左右方向键调控输入位置，可以使用Backspace和Delete键删除字符，长按可连续删除。

对话内容过多时，可以使用鼠标滚轮上下滚动，回顾过往的对话。

（补充or更改1111）

2.4.2 决策系统

Ai给一些形容词来描述蛋，让玩家选，然后用选出来的词语，自己来画一个颗蛋（我们有基础的素材，ai实现挑选素材、给素材选颜色）

限于语言AI的不完备性，AI给出的回复有时不能很好地适应格式需求。此时我们会重新将记录发给AI，尝试让AI重新生成一份合适的回复。语言AI对于图像、颜色等的处理稍逊色，但我们已经努力令其生成合适的结果，并将它选定的素材和颜色按照合适的次序，以随机数生成位置的方式组合，并呈现在玩家眼前。

实现方法 难点克服（11111）

2.5 游戏性

很明显，我们是真的想做一个体验感良好的游戏，合理的剧情、舒适的交互、清晰的规则，我们实现每一个功能都并非堆砌，是实打实地玩家更好的用户体验。

2.5.1 菜单

我们的游戏有三种菜单，分别为游戏开始时呈现的主菜单、设置菜单、暂停菜单和死亡菜单。由图片渲染、文字渲染和按钮组件组成。

所有的按钮初始化时，给定4个参数，确定按钮组件相对于窗口的大小，随后由onResize函数根据窗口大小计算出每个组件的实际大小。每个组件中有位置、宽高、定位方式、名称、浮窗描述、是否可用、颜色、默认文字颜色等属性，配备onMouseDown、onMouseUp、onHover、onTick四个回调函数。每当鼠标传入操作时，系统判断鼠标的位置是否位于这些组件之上，然后根据鼠标操作调用组件相应的回调函数，实现事件响应。

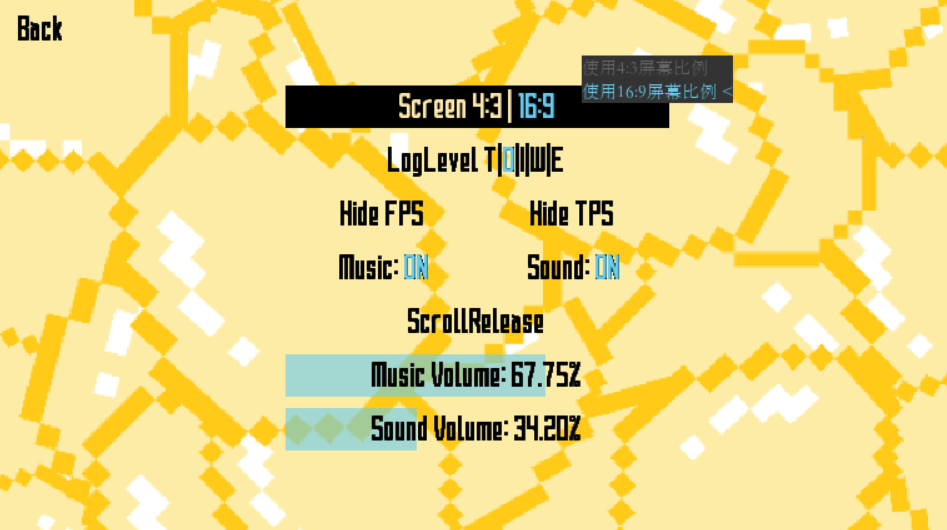
 

图 左（主菜单）右（设置菜单）

图 左（暂停菜单）右（死亡画面）

2.5.2 BGM

主菜单有一个音乐，每个世界有一个音乐（有一个音乐我听的耳朵快起茧子了），捡米粒织窝都有各自的音效。

为了保证音乐顺利播放，我们通过统一的音乐管理器实现。

背景音乐和音效被分开管理，因为考虑到我们对两者有明确不同的需求：一个一直播放，游戏开始到游戏结束之间始终播放，可能会随世界变化而变化，但不停；另一个随玩家的操作播放或暂停。

但实现的方式大同小异。各有一个表示开关的bool类、表示音量大小的float类（我们的音量可在setting里拖拽调整）、存有可能会播放的所有音乐的列表，以及播放状态列表。如我们为了实现暂停存档，播放状态列表会存放暂定的时间，没有暂停过就是0。音效，因为pygame.mixer.Sound用法与music不同，music一个时间只能有一个声音在播，sound可多个同播，则sound状态列表放置在播的音效，以便于对任意一个精准执行暂停的命令。

2.6 代码

由于我们的代码基底搭建时间较早，因而没有完全采用提供的模板。参考模板，我们最终对基底做了较大的扩展和修改。

1. 游戏很大程度上仿照了Minecraft的代码结构和术语名称，例如Block方块即Tile；Entity实体即所有不可动实体、NPC和Player，World世界即Scene
2. 采用多线程，分事件、渲染、游戏、异步四线程，分别处理玩家输入消息、渲染、游戏更新、AI交互，提升游戏性能。
3. 舍弃了原有的EventListener机制，改为tick机制、信息取用机制（而不是直接响应），实现次序更严谨事件处理逻辑；将交互信息存储在Interact中，在Window、World等不同位置均可以获取所有的玩家交互信息，让玩家拥有更丰富的操作，和更简便的代码实现。
4. 不需要Collidable类；所有元素均可在tick中实现其自身的碰撞逻辑，主要是MoveableEntity。
5. 所有Box类改为Window类，并定义了更丰富的组件和功能。
6. Portal类，改为世界上的一个“传送法阵”方块。
7. 和Java源文件结构，减少单文件的代码量和类的数量，并按不同功能将代码分装在各个文件和各个文件夹中。
8. 多态思维、更深度地面向对象，扩展了各个游戏元素的表现和行为丰富度。

以下是文件结构：

- assets/ 所有静态图像、声音资源

- block/ 方块相关逻辑

  - block.py 所有的方块类和子类

  - manager.py 方块资源管理器，将方块类和方块ID一一对应，减少循环import、局部import

- entity/ 实体相关逻辑

  - active\_skill.py 所有主动技能类

  - enemy.py 所有敌对生物类

  - entity.py 实体基类，及玩家、蛋等末端实体类

  - manager.py 实体资源管理器，将实体类和实体ID一一对应，减少循环import、局部import

  - skill.py 技能基类及所有被动技能类- interact/ 交互相关逻辑

  - \_\_init\_\_.py 所有玩家交互信息（因为久远的原因放在了\_\_init\_\_.py）

  - status.py 状态类，用于保存和简化处理玩家交互信息

- LLA/ AI交互逻辑

- music/ 声音、音效

  - music.py 音像资源管理器

- render/ 渲染逻辑

  - font.py 管理所有字体资源

  - renderable.py 渲染基类，及所有可渲染对象（因为久远实际可弃用）

  - renderer.py 渲染器

  - resource.py 管理纹理图片资源

- save/ 存档逻辑

  - configs.py 处理游戏配置文件，保存玩家设置

  - save.py 处理游戏存档数据

- user/ 玩家信息。由游戏自动生成，首次运行前不存在

  - archive/ 所有存档文件

  - config.json 游戏配置文件

- utils/ 所有工具模块和类工具模块

  - \_\_init\_\_.py 日志、报错信息优化（因为久远的原因放在了\_\_init\_\_.py）

  - element.py 游戏元素基类。与Item协作，现可弃用

  - error.py 游戏内定义的错误类

  - game.py 游戏框架逻辑，游戏管理器

  - sync.py 用于防止多线程数据冲突造成游戏进行不协调

  - text.py RenderableString类，简化文本渲染流程

  - vector.py 向量类，提高位置计算相关代码可读性和流程

- window/ 窗口相关逻辑

  - hud.py HUD类，在游戏中实时现实游戏信息，包括血条、成长值、文字提示等

  - ingame.py 游戏内窗口，包括状态窗口、任务窗口等

  - input.py 输入窗口，主要包含AI助手窗口

  - widget.py 窗口按钮

  - window.py 窗口基类，及开始窗口等游戏流程外窗口；鼠标浮窗

- world/ 游戏世界（即场景）相关逻辑

  - world.py 所有世界（场景）类

- main.py 游戏入口点

- test\_code.py 用于临时测试、单元测试

3 创意

3.1剧情窗口

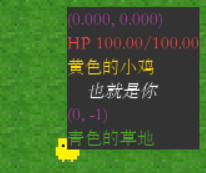
为了增加玩家沉浸式的体验，游戏设置了剧情窗口，通过游戏开始前的剧情介绍，让玩家沉浸式进入游戏的世界。在剧情窗口中，玩家可以通过按钮与窗口互动，逐页翻动剧情窗口或直接跳过。

3.2动画

利用游戏每一帧都会渲染，让相同的实体在不同帧按顺序渲染不同的图片，从而实现了动画感。无法在文档中展示，可以在游戏中的织窝和教化时看到哦。

3.3 鼠标浮窗

在所有菜单界面，将鼠标悬浮在按钮上方时，按钮底色及文字颜色会进行对应更改，增强菜单的交互性。

3.4 实时战斗系统

玩家小鸡的技能大赏:

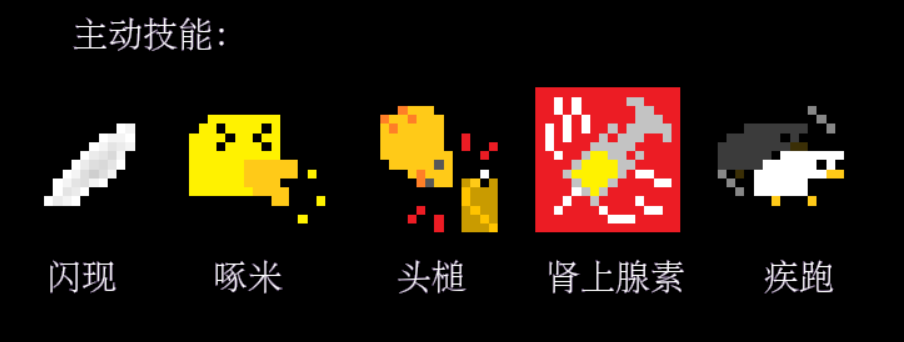


图 主动技



图 被动技

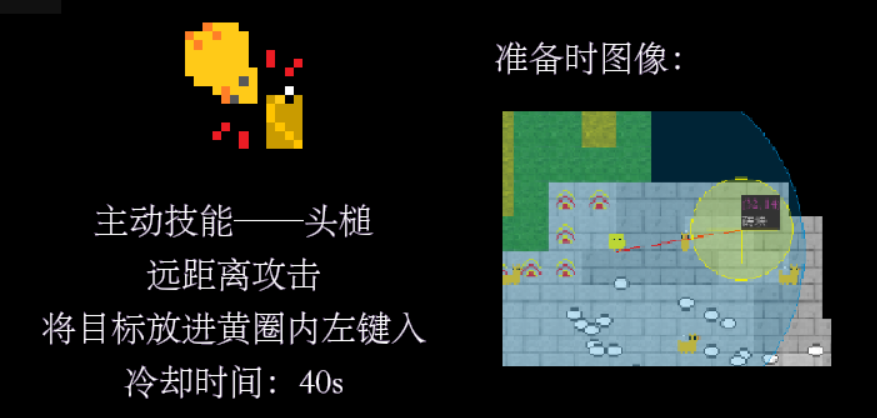


图 其中一个技能

为了增强游戏性和趣味性，我们采用了实时战斗模式。玩家独享一个主动技能列表、一个被动技能列表，分别存储玩家拥有的主动/被动技能。由于技能机制的特殊性，此模块采用了事件响应的机制，在玩家受到伤害前后、tick执行前后、成长前后、死亡前添加了事件队列。技能的获得是永久的，获得技能后，技能会在玩家的相应事件队列中永久地添加响应器来实现技能功能。例如被动技能屹立不倒，就是在preDeath死亡前队列添加响应器取消死亡，然后为玩家回复一定的生命值。主动技能都拥有onUse，由Player类的tick函数检测玩家的鼠标左键行为，调用选中主动技能的onUse触发主动技能。

3.5 个性化设置窗口大小

考虑到pygame的渲染分辨率问题和玩家的电脑屏幕大小不同，初始窗口的大小一定是玩家电脑屏幕长款的各一半；随后允许玩家手动更改窗口的大小，以获取更佳的游戏视觉体验。我们对游戏做了16:9和4:3两种比例的适应，玩家可以在设置中调节选用的屏幕比例。

为了实现不同比例的渲染，游戏采用渲染器Renderer对pygame的渲染再封装，给每个Texture一个固定的缩放比例，给Renderer一个适应窗口大小的可变缩放比例，并将所有的渲染任务全部交由Renderer完成，窗口大小改变时统一重算表面缩放并保存，避免了每帧计算缩放导致的严重性能问题。

3.6本地保存功能

考虑新手玩家的游戏流程不熟悉且前期难度较大，或是玩家希望保存当前生成的世界，游戏配备了保存的功能。使用python的json库，将World中所有的Block，Entity（包括NPC，Player以及Player的技能）打包为字典保存在json文件中，玩家在存档加载界面就可以看到玩家先前保存的存档，存档过多时还可以上下滚动鼠标滚轮翻看。

3.7随机数种子指定、世界名称指定

为了适应存档功能，游戏允许玩家在创建世界时指定游戏随机数种子和存档名称（也即世界名称）。留空时，随机数种子采取python的time库perf\_counter\_ns()作为游戏种子，名称默认为“种子+序列世界”。如果玩家指定的种子不合法，或世界名称已存在，游戏会给出提示并拒绝玩家创建世界。极小概率下，如果默认名称重复，则在名称后添加“（1）”“（2）”确保一定不存在重复名称。

他说还有一个Consistency?最后的20分